FRONT-END FORMATIONS

Level 5 - Fonts & Interactions

TABLE OF CONTENTS

TABLE OF CONTENTS

FONT FACE

Using @font-face, we have the ability to provide online fonts for use on our websites.

FONT FACE

Example usage of @font-face:

```
@font-face {



}
```

FONT FACE

We specify the font-family, which is what we'll use to call the font:

```
@font-face {
  font-family: 'OpenSansRegular';




}
```

FONT FACE

We add the location of the font files through the src property:

```
@font-face {
  font-family: 'OpenSansRegular';
  src: url('OpenSansRegular-webfont.eot');



}
```

FONT FACE

We add the location of the font files through the src property:

```
@font-face {
  font-family: 'OpenSansRegular';
  src: url('OpenSansRegular-webfont.eot');



}
```

we'll have to specify multiple font types, which can be added as additional url()'s to the files.

FONT FACE

We specify the font-style:

```
@font-face {
  font-family: 'OpenSansRegular';
  src: url('OpenSansRegular-webfont.eot');
  font-style: normal;


}
```

FONT FACE

We specify the font-weight:

```
@font-face {
  font-family: 'OpenSansRegular';
  src: url('OpenSansRegular-webfont.eot');
  font-style: normal;
  font-weight: normal;
}
```

FONT FACE

Using @font-face in our stylesheet:

```
h1 {
  font-family: 'OpenSansRegular';
}
```

FONT FACE

Using @font-face in our stylesheet:

```
h1 {
  font-family: 'OpenSansRegular';
}
```

we specify the font-family as the same one established in the @font-face call.

FONT FACE

With @font-face fonts, just like any other font declaration, we'll want to add fallback fonts.

FONT FACE

Using @font-face in our stylesheet with fallbacks:

```
h1 {
  font-family: 'OpenSansRegular', Helvetica, Arial, sans-serif;
}
```

Provide fallback fonts here,
as you normally would.

FONT FACE

Using varying weights with @font-face:

```
@font-face {
  font-family: 'OpenSansBold';
  src: url('OpenSansBold-webfont.eot');
  font-style: normal;
  font-weight: normal;
}
```

FONT FACE

Using varying weights with @font-face:

```
@font-face {
  font-family: 'OpenSansBold';
  src: url('OpenSansBold-webfont.eot');
  font-style: normal;
  font-weight: normal;
}
```

we're using a bold font family of 'openSansBold'.

FONT FACE

Using varying weights with @font-face:

```
h1 {
  font-family: 'OpenSansBold';
}
```

we use the bold version by changing the font-family.

FONT FACE

We can alter the @font-face call in order to use the font-weight and font-style properties as usual.

FONT FACE

Using varying weights with @font-face:

```
@font-face {
  font-family: 'OpenSansBold';
  src: url('OpenSansBold-webfont.eot');
  font-style: normal;
  font-weight: normal;
}
```

FONT FACE

Using varying weights with @font-face:

```
@font-face {
  font-family: 'OpenSansRegular';
  src: url('OpenSansBold-webfont.eot');
  font-style: normal;
  font-weight: normal;
}
```

we can instead change the font-family to the same name as the regular weight version.

FONT FACE

Using varying weights with @font-face:

```
@font-face {
  font-family: 'OpenSansRegular';
  src: url('OpenSansBold-webfont.eot');
  font-style: normal;
  font-weight: normal;
}
```

We keep the src url() the same in order to include the bold font weight.

FONT FACE

Using varying weights with @font-face:

```
@font-face {
  font-family: 'OpenSansRegular';
  src: url('OpenSansBold-webfont.eot');
  font-style: normal;
  font-weight: bold;
}
```

We change the font-weight to bold.

FONT FACE

Using varying weights with @font-face:

```
h1 {
  font-weight: bold;
}
```

we use the bold version by changing the font-weight instead of the font-family.

TABLE OF CONTENTS

TABLE OF CONTENTS

TRANSFORM

Using the transform property in CSS3, we can translate, rotate, scale, and skew elements in CSS.

TRANSLATE

You can create a 2D translation using transform:

```
.element {
  transform: translate(20px, 30px);
}
```

TRANSLATE

You can create a 2D translation using transform:

```
.element {
  transform: translate(20px, 30px);
}
```

Translate the .element 20px to the right (x-axis).

TRANSLATE

You can create a 2D translation using transform:

```
.element {
  transform: translate(20px, 30px);
}
```

Translate the .element 30px down (y-axis).

TRANSLATE

Example output of the transform translate:

TRANSLATE

Example usage of a 2D translation using transform:

```
translate(<tx>, <ty>)
```

TRANSLATE

Example usage of a 2D translation using transform:

```
translate(<tx>, <ty>)
```

A <transition-value> for the x-axis, which can be either a length or percentage.

TRANSLATE

Example usage of a 2D translation using transform:

```
translate(<tx>, <ty>)
```

A <transition-value> for the y-axis, which can be either a length or percentage. If not specified, the value is 0.

TRANSLATE

You can use translateX and translateY to translate the x and y values individually:

```
.element {
  transform: translateX(20px);
}
```

```
.element {
  transform: translateY(30px);
}
```

TRANSLATE

You can use translateX and translateY to translate the x and y values individually:

```
translateX(<tx>)
```

```
translateY(<ty>)
```

ROTATE

With rotate, you can rotate an element clockwise around its origin by the specified angle.

ROTATE

Example usage of rotate:

```
.element {
  transform: rotate(45deg);
}
```

ROTATE

Example usage of rotate:

```
.element {
  transform: rotate(45deg);
}
```

The element is rotated 45 degrees.

ROTATE

Example output of the transform rotate:

SCALE

With scale, you can do a 2D scale by a specified unitless number:

```
.element {
  transform: scale(1.2);
}
```

SCALE

With scale, you can do a 2D scale by a specified unitless number:

```
.element {
  transform: scale(1.2);
}
```

The element is scaled to the unitless number, 1.2.

SCALE

Example output of the transform scale:

SCALE

Exampled usage of scale:

```
scale(<sx>, <sy>)
```

SCALE

Exampled usage of scale:

```
scale(<sx>, <sy>)
```

A unitless number for the x-axis.

SCALE

Exampled usage of scale:

```
scale(<sx>, <sy>)
```

?

A unitless number for the y-axis. If not specified, it defaults to the value of <sx>.

SCALE

You can use scaleX and scaleY to translate the x and y values individually:

```
.element {
  transform: scaleX(1.2);
}
```

```
.element {
  transform: scaleY(0.3);
}
```

SCALE

You can use scaleX and scaleY to scale the x and y values individually:

```
scaleX(<sx>)
```

```
scaleY(<sy>)
```

SKEW

With skew, an element is skewed around the x or y axis by the angle specified.

SKEW

Example usage of skewX:

```
.element {
  transform: skewX(-25deg);
}
```

SKEW

Example usage of skewX:

```
.element {
  transform: skewX(-25deg);
}
```

The element is skewed -25 degrees along the x-axis.

SKEW

Example output of the transform skewX:

SKEW

Example usage of skewX:

```
skewX(<ax>)
```

SKEW

Example usage of skewX:

```
skewX(<ax>)
```

An <angle>
for the x-axis.

SKEW

Example usage of skewY:

```
skewY(<ay>)
```

SKEW

Example usage of skewY:

```
skewY(<ay>)
```

An <angle> for the y-axis.

SKEW

Example usage of skewX and skewY:

```
.element {
  transform: skewX(25deg);
}
```

```
.element {
  transform: skewY(-85deg);
}
```

SKEW

Example output of the transform skewX and skewY:

TABLE OF CONTENTS

TABLE OF CONTENTS

TRANSITION

CSS3 provides transitions, which allow you to transition between two states of a specified element.

TRANSITION

Example usage of transition:

```
.element {
  background-color: black;

}
```

TRANSITION

Example usage of transition:

```
.element {
  background-color: black;

}
```

```
.element:hover {
  background-color: blue;
}
```

TRANSITION

Example usage of transition:

```
.element {
  background-color: black;
  transition: background-color 0.2s ease-in-out;
}
```

```
.element:hover {
  background-color: blue;
}
```

TRANSITION

Example usage of transition:

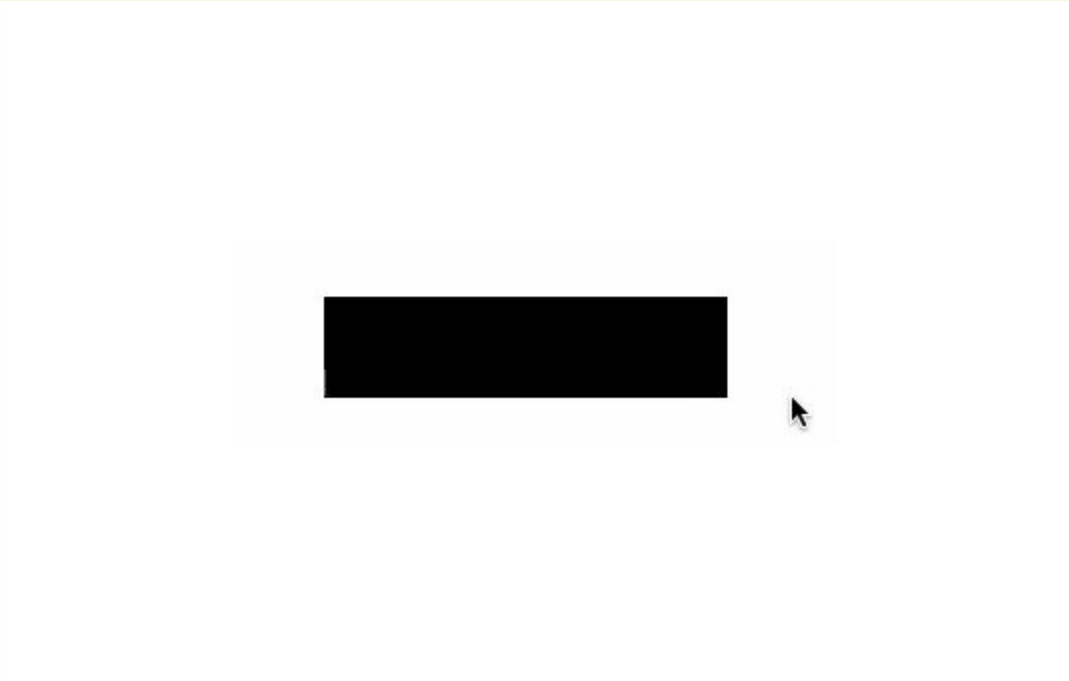
```
.element {
  background-color: black;
  transition: background-color 0.2s ease-in-out;
}
```

```
.element:hover {
  background-color: blue;
}
```

The background-color transitions from black to blue over the period of 0.2 seconds.

TRANSITION

Example output of the transition:



TRANSITION

Example usage of the shorthand transition property:

```
transition: <property> <duration> <timing-function> <delay>
```

TRANSITION

Example usage of the shorthand transition property:

```
transition: <property> <duration> <timing-function> <delay>
```

The CSS property you want to transition.

TRANSITION

Example usage of the shorthand transition property:

```
transition: <property> <duration> <timing-function> <delay>
```

The duration of the transition. The default value is 0s, or 0 seconds.

TRANSITION

Example usage of the shorthand transition property:

```
transition: <property> <duration> <timing-function> <delay>
```

The timing of the transition itself.

TRANSITION

Example usage of the shorthand transition property:

```
transition: <property> <duration> <timing-function> <delay>
```

The timing of the transition itself

TIMING-FUNCTIONS

- ease
- ease-in
- ease-in-out
- linear
- cubic-bezier
- step-start
- step-end
- steps()

TRANSITION

Example usage of the shorthand transition property:

```
transition: <property> <duration> <timing-function> <delay>
```

The amount of time to wait between the change that is being requested on a specific property, and the start of the transition.

TRANSITION

You can set the transition values individually, as well:

```
.element {



}
```

TRANSITION

You can set the transition values individually, as well:

```
.element {
  transition-property: background-color;




}
```

TRANSITION

You can set the transition values individually, as well:

```
.element {
  transition-property: background-color;
  transition-duration: 0.2s;



}
```

TRANSITION

You can set the transition values individually, as well:

```
.element {
  transition-property: background-color;
  transition-duration: 0.2s;
  transition-timing-function: ease-in-out;

}
```

TRANSITION

You can set the transition values individually, as well:

```
.element {
  transition-property: background-color;
  transition-duration: 0.2s;
  transition-timing-function: ease-in-out;
  transition-delay: 0.1s;
}
```

TRANSITION

Using all as the transition-property, we can transition multiple properties at once.

TRANSITION

Example usage of transition using the all property:

```
.element {
  background-color: black;
  color: white;

}
```

```
.element:hover {
  background-color: grey;
  color: black;
}
```

TRANSITION

Example usage of transition using the all property:

```
.element {
  background-color: black;
  color: white;
  transition: all 0.2s ease-in-out;
}
```

```
.element:hover {
  background-color: grey;
  color: black;
}
```

TRANSITION

Example usage of transition using the all property:

```
.element {
  background-color: black;
  color: white;
  transition: all 0.2s ease-in-out;
}
```

```
.element:hover {
  background-color: grey;
  color: black;
}
```

The all property will transition both the background-color AND the color properties.

TRANSITION

Example output of the transition using the all property:

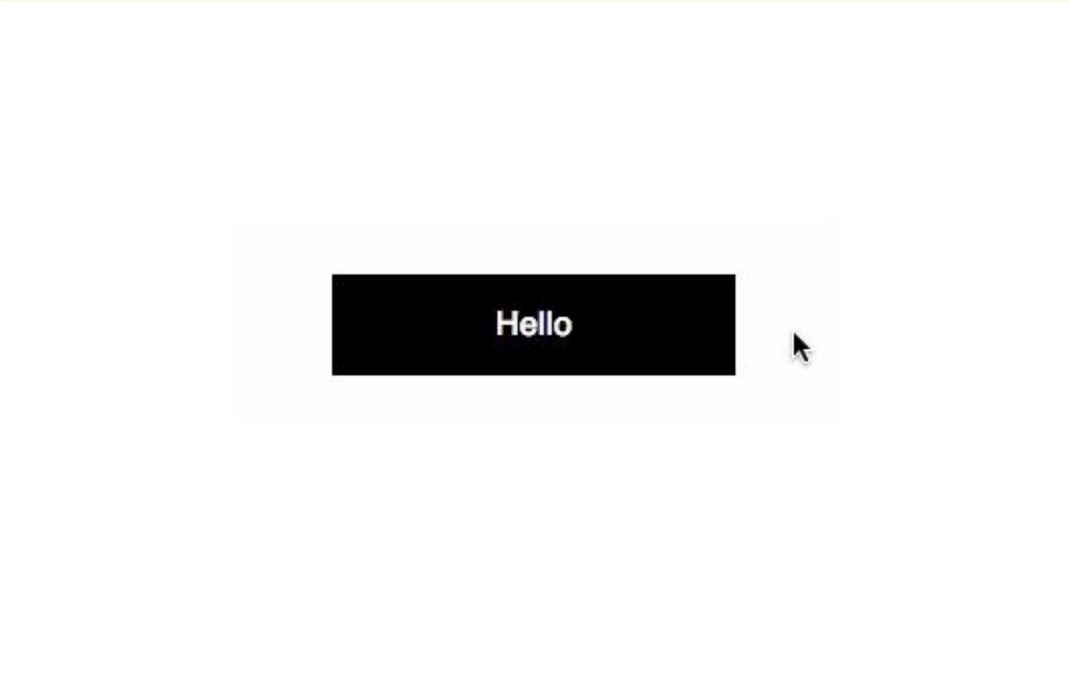


TABLE OF CONTENTS

TABLE OF CONTENTS

PROGRESSIVE ENHANCEMENT

The term “progressive enhancement” refers to the use of newer features that add to the experience in modern browsers that support those features, but doesn’t detract from the experience in older browsers.

PROGRESSIVE ENHANCEMENT

Example of progressive enhancement:

```
.element {
  background: #ccc;
  border-radius: 10px;
  box-shadow: 0 1px 1px rgba(0, 0, 0, 0.75);
}
```

PROGRESSIVE ENHANCEMENT

Example of progressive enhancement:

```
.element {
  background: #ccc;
  border-radius: 10px;
  box-shadow: 0 1px 1px rgba(0, 0, 0, 0.75);
}
```

PROGRESSIVE ENHANCEMENT

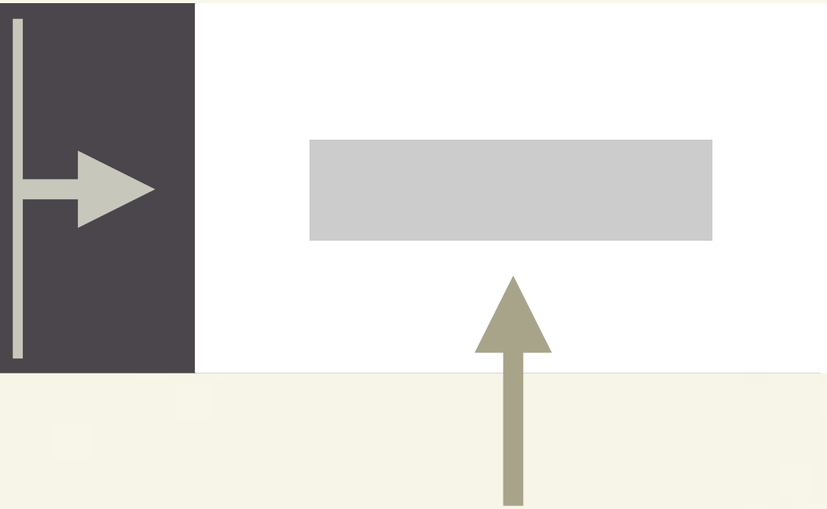
Example of progressive enhancement:

```
.element {
  background: #ccc;
  border-radius: 10px;
  box-shadow: 0 1px 1px rgba(0, 0, 0, 0.75);
}
```

PROGRESSIVE ENHANCEMENT

Example of progressive enhancement:

```
.element {
  background: #ccc;
  border-radius: 10px;
  box-shadow: 0 1px 1px rgba(0, 0, 0, 0.75);
}
```



If the border-radius and box-shadow properties aren't supported, we still get a usable design.

FRONT-END FORMATIONS